Guide to Using GPUs and MPI on Pleiades

Thomas Nabelek, NASA LaRC HPC Incubator, Last Updated July 25, 2017

This guide will assist you in learning how to access and use the resources needed to compile and run OpenACC GPU, MPI, and hybrid OpenACC/MPI code on the Pleiades computing resources. The instructions given will walk you through compiling and running the example code in an interactive node, and then running the code with the same resources using a job file.

The example Laplace problem code used here is from the 2017 XSEDE HPC Summer Boot Camp by Pittsburgh Supercomputing Center: psc.edu/136-users/training/2561-xsede-hpc-workshop-june-6-9-2017-summer-boot-camp. The C code has been modified so that it runs without user input. Download the files from sites.larc.nasa.gov/hpc/files/2016/05/pleiades_tutorial.zip to your local machine.

The commands below must be run in a UNIX command prompt.

Run the commands in the gray boxes and blue boxes to compile and execute the example OpenACC code using GPUs.

Run the commands in the gray boxes and green boxes to compile and execute the example MPI code.

Run the commands in the gray boxes and yellow boxes to compile and execute the example hybrid MPI and OpenACC code.

Documentation

The NASA Advanced Supercomputing (NAS) Division's High-End Computing Capability (HECC) site provides very useful documentation at nas.nasa.gov/hecc/support/kb. Some particularly useful documents include:

Pleiades Specs
nas.nasa.gov/hecc/resources/pleiades.html

New User Orientation
nas.nasa.gov/hecc/support/kb/161

Logging Into NAS Systems for the First Time
nas.nasa.gov/hecc/support/kb/logging-into-nas-systems-for-the-first-time_539.html

Inbound File Transfer Through SFEs: Examples
nas.nasa.gov/hecc/support/kb/inbound-file-transfer-through-sfes-examples_144.html

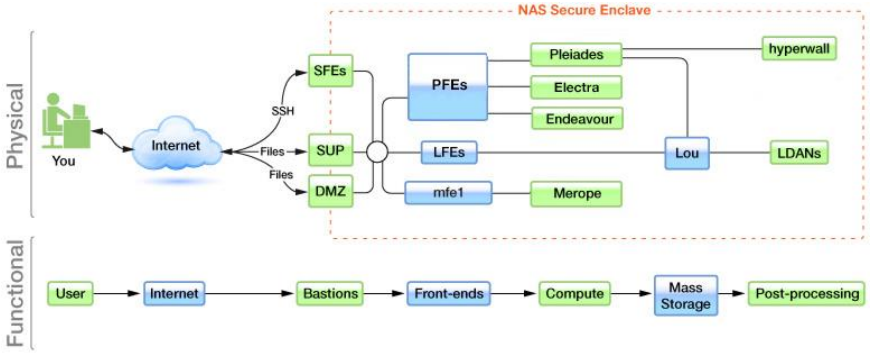
Pleiades Job Queue Structure
nas.nasa.gov/hecc/support/kb/pleiades-job-queue-structure_187.html

Using GPU Nodes
nas.nasa.gov/hecc/support/kb/using-gpu-nodes_298.html

The Lou Mass Storage System
nas.nasa.gov/hecc/support/kb/the-lou-mass-storage-system_371.html

Logon to Pleiades

Users will need their NAS password, PIN, and token generator (app or fob) before proceeding. Users must first ssh into the NAS Secure Front-End (SFE) before connecting to the Pleiades Front-End (PFE).



NAS System Structure, from nas.nasa.gov/hecc/support/kb/hpc-environment-overview_25.html

Replace 'X' with '1', '2', or '3'. If the user's NAS username is different than his or her AUID, 'username@' must be placed before 'sfe':

```
ssh sfeX.nas.nasa.gov
```

When you get to the 'PAM Authentication Enter PASSCODE:' prompt, enter the passcode/token given by your token generator. If you are using a soft token generator (the RSA Token app), you must first enter your 8-digit PIN. Note that if the PIN is entered incorrectly, you will not be notified and will be shown an invalid token. After you have entered the passcode/token, you will be prompted for your NAS password. This may be different from your NDC or Launchpad password.

Once on the SFE, do:

```
ssh pfe
```

At the prompt, you will need to enter your NAS password again. This two-step login process can be simplified by setting up SSH Passthrough.

Once you have logged on to a front-end node, you can submit a job using a job file, or request an interactive node to do some setup and testing. Instructions on how to do this are given below. Do not run jobs directly on the front-end login node. The default shell on Pleiades is bash.

Copying Files to Pleiades

Copying files to Pleiades is a two-step process similar to the login process. One must first copy files to the SFE system and then copy them from the SFE system to the PFE. Again, this process can be simplified by setting up SSH Passthrough. `scp` is the command most often used to copy files. See nas.nasa.gov/hecc/support/kb/inbound-file-transfer-through-sfes-examples_144.html for more details.

To transfer the example files, **open a different command prompt**, and do:

```
cd path_to_directory_containing_example_files
      e.g.: cd ~/Downloads

mv name_of_example_files pleiades_tutorial.zip
      e.g.: mv pleiades_tutorial_rev_C.zip pleiades_tutorial.zip

scp pleiades_tutorial.zip sfe3.nas.nasa.gov:~
```

Enter your login PASSCODE and NAS password. Then, **at the command prompt where you are logged into Pleiades**:

```
cd ~

scp sfe3:~/pleiades_tutorial.zip .
      If you get an error here saying 'Permission denied (keyboard-interactive)', try again.

unzip pleiades_tutorial.zip

rm pleiades_tutorial.zip

cd pleiades_tutorial
```

You should now be in the directory containing the example files, located in your home directory on the Pleiades system.

Note that files copied to sfe3 will only be available on sfe3, and not on sfe1 or sfe2. The same is true for sfe1 and sfe2.

Job Submission, Scheduling, and Queues

From nas.nasa.gov/hecc/support/kb/portable-batch-system-(pbs)-overview_126.html:

> All NAS facility supercomputers use the Portable Batch System (PBS) from Altair for batch job submission, job monitoring, and job management.
>
> Batch Jobs
>
> Batch jobs run on compute nodes, not the front-end nodes. A PBS scheduler allocates blocks of compute nodes to jobs to provide exclusive access. You will submit batch jobs to run on one or more compute nodes using the `qsub` command from an interactive session on one of Pleiades front-end systems (PFEs).
>
> Normal batch jobs are typically run by submitting a script. A "jobid" is assigned after submission. When the resources you request become available, your job will execute on the compute nodes. When the job is complete, the PBS standard output and standard error of the job will be returned in files available to you.
>
> Queues
>
> The available queues on different systems vary, but all typically have constraints on the maximum walltime and/or the number of CPUs allowed for a job. Some queues may also have other constraints or be restricted to serving certain users or groups. In addition, to ensure that each NASA mission directorate is granted their allocated share of resources at any given time, mission directorate limits (called "shares") are also set on Pleiades.

Additionally, each queue has different processing elements available to it. For example, the 'normal' queue has Broadwell, Haswell, Ivy Bridge, and Sandy Bridge nodes available while the 'gpu_k40' queue has nodes with Sandy Bridge CPUs and Tesla K40 GPUs attached. More information on queues and scheduling can be found at nas.nasa.gov/hecc/assets/pdf/training/PBS_Queues_and_Scheduling.pdf.

View Queue Information

View status of specific job:

```
qstat job_id*
```

View all jobs submitted by user:

```
qstat -u username
```

View jobs submitted to GPU queue:

```
qstat queue_name
```

```
qstat gpu_k40
```

```
qstat normal
```

```
qstat gpu_k40
```

Get information about all queues:

```
node_stats.sh
```

Get list of all queues and information:

```
qstat -q
qstat -Q
```

Get reason why job is waiting in queue:

```
qstat -as job_id*
```

Get job history for user:

```
qstat -u username -x
```

* The job ID is the number listed at the beginning of the line for a given job when `qstat -u username` is run.

Request An Interactive Node

No code should be run on the login node. You must request a compute node to run a job. Nodes can be requested either using a job file (described below) or in an interactive mode. Requesting an interactive mode is most useful when compiling, debug, and testing out code. To request an interactive node, do:

```
qsub -I -l select=num_nodes[:ncpus=num_cores_per_node]:model=node_arch -q
      queue_name -l walltime=hh:mm:ss
```

```
qsub -I -l select=1:model=san_gpu -q gpu_k40 -l walltime=00:30:00
```
This will request 1 node from the GPU queue with a wall time of 30 minutes.

```
qsub -I -l select=4:ncpus=1:mpiprocs=1:model=ivy -q normal -l
walltime=00:30:00
```
This will request 4 nodes, 1 CPU core/node, 1 MPI process/node, using the Ivy Bridge architecture from the normal queue with a wall time of 30 minutes.

```
qsub -I -l select=4:mpiprocs=1:model=san_gpu -q gpu_k40 -l walltime=00:30:00
```
This will request 4 nodes with 1 MPI process/node on the GPU queue with a wall time of 30 minutes.

Depending on the availability of nodes in the queue that you are accessing, the wait time to be placed on a node could be a while. At the time of writing, access to a GPU node is generally granted within one minute. The wait time on the normal queue can be much longer.

Options that can be used with the `-l` argument include but are not limited to the following:

select – number of nodes

ncpus – number of CPU cores per node

mpiprocs – number of MPI processes per node

model – node architecture

mem – memory requirement

Note that when you are granted access to a node, you have exclusive access to that node regardless of how many cores you requested.

Manage Modules

On Pleiades, like many other HPC systems, software and the runtime environment necessary for that software are managed using modules. Because different packages, especially different versions of the same software, can conflict with one another when it comes to required executables and libraries, each module is activated and deactivated as needed.

nas.nasa.gov/hecc/support/kb/using-software-modules_115.html:

`module avail` – get list of available modules

`module load` *`module_name`* – load desired module

`module list` – get list of loaded modules

`module unload` *`module_name`* – unload desired module

`module show` *`module_name`* – see changes that will occur in the environment if you load *module_name*

`module load comp-pgi`
This will load the comp-pgi module to make the pgcc compiler and necessary libraries available.

`module load comp-intel mpi-sgi`
This will load the modules needed to setup the environment for compiling and running MPI code.

`module load comp-pgi comp-intel mpi-intel`
This will load the modules needed to setup the environment for compiling and running hybrid MPI and OpenACC code.

Note: Leaving off the version number of the module will load the default version of that module, e.g. 'module load comp-pgi' will load the default version of the comp-pgi module (comp-pgi/17.1 at the time of writing).

For the sake of keeping build and run environments from breaking, users should be consistent with which version of a particular module is used. For production codes, if a particular module version has been verified to work, that particular version should be explicitly loaded in the job file (discussed below) rather than relying on the default version loaded to always function as expected. For similar reasons, codes should often be run under the same environment (same modules loaded) as they were compiled under.

Compile Code

Compile the example OpenACC GPU code:

C code: `pgcc -acc -ta=tesla,cuda8.0 laplace_acc.c -o laplace_acc.out`
Fortran code: `pgf90 -acc -ta=tesla,cuda8.0 laplace_acc.f90 -o laplace_acc.out`

For comparison, compile the example serial code:

C code: `pgcc laplace_serial.c -o laplace_serial.out`
Fortran code: `pgcc laplace_serial.f90 -o laplace_serial.out`

Documentation for the pgcc compiler can be found at geco.mines.edu/guide/pgcc.html and geco.mines.edu/prototype/How_do_you_build_applications/pg/pgcc.html. The `-Minfo` flag can output useful information to stderr.

Compile the example MPI code:

C code: `icc -lmpi laplace_mpi.c -o laplace_mpi.out`
Fortran code: `ifort -lmpi laplace_mpi.c -o laplace_mpi.out`

For comparison, compile the example serial code:

C code: `icc laplace_serial.c -o laplace_serial.out`
Fortran code: `ifort laplace_serial.c -o laplace_serial.out`

Compile the hybrid MPI and OpenACC C code:

```
pgcc -acc -lmpi -I/nasa/mvapich2/2.1/intel/include
-L/nasa/mvapich2/2.1/intel/lib laplace_hybrid.c -o laplace_hybrid.out
```

For comparison, compile the example serial code:

```
pgcc laplace_serial.c -o laplace_serial.out
```

Run Code

```
./laplace_acc.out
```

```
mpiexec -np 4 ./laplace_mpi.out
```

```
mpirun -np 4 ./laplace_hybrid.out
```

```
./laplace_serial.out
```

Be sure to leave the executables in the tutorial directory. They will be used by the job files submitted below.

Submitting Batch Jobs Using Job Files

Job files are much like UNIX shell scripts, but include PBS directives to request resources and specify job options. Compare the provided example job files to the interactive node request commands given above. You will see that the `qsub` arguments are simply moved into the job file as PBS directives. An additional PBS directive not given in the example above is the `-m` flag: The `-m` flag can be used to setup email alerts pertaining to your job status. Use `a` to be alerted when the job is aborted by the batch system, `b` to be alerted when the job is beginning execution, and `e` to be alerted when the job terminates.

`qsub` *`jobfile`*

If you are still in an interactive node, first logout:

```
logout
```

You should now be back on a Pleiades Front-End node.

```
qsub tutorial_acc.job
```

```
qsub tutorial_mpi.job
```

```
qsub tutorial_hybrid.job
```

Use the qstat commands given above to monitor the submitted job. Once it is complete, you should have tutorial_acc.stdout and tutorial_acc.stderr files, tutorial_mpi.stdout and tutorial_acc.stderr files, or tutorial_hybrid.stdout and tutorial_hybrid.stderr files, with output and errors in the ~/pleiades_tutorial/ directory.

- A /nobackup directory on a Lustre filesystem, which you can use to temporarily store larger files for reading and writing large amounts of data while running jobs

For long-term data storage, you also have access to a home directory on the Lou mass storage systems. The /nobackup filesystems are mounted on Lou, so you can easily copy files there directly from your /nobackup directory.

The Pleiades and Lou home filesystems are backed up each night. These backups are stored for approximately one year. The scratch (/nobackup) filesystems are *not* backed up.

Quota limits are enforced on all filesystems. Two kinds of quotas are supported:

- Limits on the total disk space occupied by your files
- Limits on the number of files you can store, irrespective of size; for quota purposes, directories count as files